S/N 09/819022

#4

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Iivonen et al		
Serial No.:	09/819022	Group Art Unit:	2171
Filed:	3/26/01	Docket No.:	796.384USW1
Title:	COMPRESSION OF NODES IN A TRIE STRUCTURE		

CERTIFICATE UNDER 37 C.F.R. 1.8: The undersigned hereby certifies that this Transmittal Letter and the paper, as described herein, are being deposited in the United States Postal Service, as first class mail, with sufficient postage, in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on 26 June 2001.

Michael B. Lasky
Name

Signature

SUBMISSION OF PRIORITY DOCUMENT

Box Missing Parts
Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

Enclosed is a certified copy of Finnish application, Serial Number 982095, filed 29 September 1998, the priority of which is claimed under 35 U.S.C. §119.

Respectfully submitted,

Altera Law Group, LLC
6500 City West Parkway
Suite 100
Minneapolis, MN 55344-7701
(952) 912-0527

Date: 26 June 2001

By: _____
Michael B. Lasky
Reg. No. 29,555
MBL/vlb

PATENTTI- JA REKISTERIHALLITUS
NATIONAL BOARD OF PATENTS AND REGISTRATION

Helsinki 20.3.2001

E T U O I K E U S T O D I S T U S
P R I O R I T Y D O C U M E N T

Hakija Applicant	**Nokia Telecommunications Oy** **Helsinki**
Patenttihakemus nro Patent application no	**982095**
Tekemispäivä Filing date	**29.09.1998**
Kansainvälinen luokka International class	**G06F**

Keksinnön nimitys
Title of invention

"Menetelmä muistin toteuttamiseksi ja muistijärjestely"

Hakijan nimi on hakemusdiaariin **30.01.2000** tehdyn nimenmuutoksen jälkeen **Nokia Networks Oy.**

The application has according to an entry made in the register of patent applications on **30.01.2000** with the name changed into **Nokia Networks Oy.**

Täten todistetaan, että oheiset asiakirjat ovat tarkkoja jäljennöksiä patentti- ja rekisterihallitukselle alkuaan annetuista selityksestä, patenttivaatimuksista, tiivistelmästä ja piirustuksista.

This is to certify that the annexed documents are true copies of the description, claims, abstract and drawings originally filed with the Finnish Patent Office.

Pirjo Kaila
Tutkimussihteeri

Maksu 300,- mk
Fee 300,- FIM

Osoite: Arkadiankatu 6 A, P.O.Box 1160, FIN-00101 Helsinki, FINLAND
Puhelin: 09 6939 500
Telephone: + 358 9 6939 500
Telefax: 09 6939 5328
Telefax: + 358 9 6939 5328

Menetelmä muistin toteuttamiseksi ja muistijärjestely

Keksinnön ala

Keksintö liittyy yleisesti muistin toteuttamiseen. Tarkemmin sanottuna, keksintö on tarkoitettu käytettäväksi kaikkien sellaisten muistien yhteydessä, jotka perustuvat funktionaaliseen digitaaliseen trie-rakenteeseen. Ratkaisu on kuitenkin tarkoitettu lähinnä keskusmuistitietokantoja varten. Funktionaalisella rakenteella tarkoitetaan sellaista muistia, jossa päivitykset, kuten lisäykset suoritetaan siten, että ensin kopioidaan polku juuresta lisäyskohtaan ja lisäys suoritetaan kopioituun tietoon (lisäystä ei suoriteta suoraan olemassa olevaan tietoon). Tällaista päivitysmenettelyä kutsutaan myös nimellä "copy-on-write".

Keksinnön tausta

Keksinnön mukainen periaate pohjautuu sinänsä tunnettuun yksiulotteiseen hakemistorakenteeseen, jota kutsutaan nimellä digitaalinen trie (engl. digital trie, sana "trie" tulee englannin kielen sanasta "retrieval"). Digitaalisia trie-rakenteita voidaan toteuttaa kahden tyyppisinä: sellaisina, joissa on sankoja (bucket trie) ja sellaisina, joissa ei ole sankoja.

Sangollinen digitaalinen trie-rakenne on puumainen rakenne, jossa on kahdenlaisia solmuja: sankoja ja trie-solmuja. Sangoksi kutsutaan tässä vaiheessa tietorakennetta, johon mahtuu joukko tietoyksikköjä tai joukko osoittimia tietoyksiköihin tai joukko hakuavain/tietoyksikkö-pareja tai joukko hakuavain/osoitin-pareja. Joukolle on määritelty jokin ykköstä suurempi maksimikoko. Sanko voi kuitenkin sisältää tätä maksimikokoa pienemmän lukumäärän tietoyksikköjä, osoittimia tai avain/osoitin-pareja, jolloin sanko ei siis ole täynnä. Trie-solmu on puolestaan hakua ohjaava taulukko, jonka koko on kaksi potenssiin k (2^k) alkiota. Jos trie-solmun alkio on käytössä, se osoittaa joko hakemistopuun seuraavalla tasolla olevaan trie-solmuun tai sankoon. Muutoin alkio on vapaa (tyhjä).

Tietokantaan kohdistuva haku etenee tutkimalla hakuavainta (joka on esim. matkaviestinverkon tai puhelinkeskuksen tilaajatietokannan tapauksessa tyypillisesti se binääriluku, joka vastaa tilaajan puhelinnumeroa) k bittiä kerrallaan. Tutkittavat bitit valitaan siten, että rakenteen ylimmällä tasolla (ensimmäisessä trie-solmussa) tutkitaan k:ta eniten vasemmalla olevia bittiä, rakenteen toisella tasolla vasemmalta lukien seuraavia k:ta bittiä, jne. Tutkittavat bitit tulkitaan etumerkittömänä kaksijärjestelmän lukuna, jota käytetään suoraan

trie-solmun sisältämän alkiotaulukon indeksinä, joka osoittaa tietyn alkion taulukosta. Jos indeksin mukainen alkio ei ole käytössä, haku päättyy epäonnistuneena. Jos alkio osoittaa seuraavalla tasolla olevaan trie-solmuun, siellä tutkitaan edellä kuvatulla tavalla k:ta seuraavaa bittiä, jotka on irrotettu hakuavaimesta. Vertailun perusteella haaraudutaan trie-solmussa edelleen joko seuraavalla tasolla olevaan trie-solmuun tai sankoon. Jos alkio osoittaa avaimen sisältävään sankoon, talletettua avainta verrataan hakuavaimeen. Koko hakuavaimen vertailu tapahtuu siten vasta haun saavuttaessa sangon. Avainten ollessa samat haku on onnistunut, ja haluttu tietoyksikkö saadaan sangon osoittimen osoittamasta muistiosoitteesta. Avainten ollessa erisuuret haku päättyy epäonnistuneena.

Sangottomassa trie-rakenteessa ei ole sankoja, vaan sankoa vastaa lehtisolmu, joka sisältää vain yhden elementin, joka voi olla tietoyksikkö, osoitin tietoyksikköön, hakuavain/tietoyksikkö-pari tai hakuavain/osoitin-pari. Sangottomassa trie-rakenteessa lehtisolmujen yläpuolella olevia solmuja nimitetään tässä yhteydessä sisäsolmuiksi. Sangottomassa digitaalisessa trie-rakenteessa solmut ovat siis joko sisäsolmuja tai lehtisolmuja. Sankojen avulla saadaan hakemistorakenteen muokkaustarvetta lykättyä ajallisesti, koska sankoihin saadaan mahtumaan paljon osoittimia, tietoyksiköitä, hakuavain/tietoyksikköpareja tai hakuavain/osoitin-pareja ennen kuin kyseinen tarve syntyy.

Keksinnön mukaista ratkaisua voidaan soveltaa sekä sangolliseen että sangottomaan rakenteeseen. Jatkossa käytetään esimerkkeinä sangollisia rakenteita, koska niiden avulla myös sangoton trie-rakenne on helppo ymmärtää.

Kuviossa 1 on esitetty esimerkki digitaalisesta trie-rakenteesta, jossa avaimen pituus on 4 bittiä ja k=2, jolloin siis kussakin trie-solmussa on 2^2=4 alkiota ja kullakin tasolla tutkitaan kahta avaimesta irrotettua bittiä. Sankoja on merkitty viitemerkeillä A, B, C, D...H...M, N, O ja P. Sanko on siis sellainen solmu, josta ei enää osoiteta puun alemmalle tasolle. Trie-solmuja on kuviossa 1 merkitty viitemerkeillä IN1...IN5 ja trie-solmun alkioita viitemerkeillä NE.

Kuvion 1 mukaisessa esimerkkitapauksessa ovat esitettyjen sankojen hakuavaimet seuraavat: A=0000, B=0001, C=0010,..., H=0111,... ja P=1111. Kuhunkin sankoon on tässä tapauksessa talletettu osoitin siihen tietokannan SD muistipaikkaan, josta varsinainen data, esim. kyseisen tilaajan puhelinnumero sekä muut kyseistä tilaajaa koskevat tiedot löytyvät. Tietokannassa voi varsinainen tilaajadata olla talletettuna esim. peräkkäistiedostoksi kuviossa esi-

tettyyn tapaan. Esim. tietueen H hakuavaimen perusteella tapahtuu haku irrottamalla hakuavaimesta ensin kaksi vasemmanpuoleisinta bittiä (01) ja tulkitsemalla ne, jolloin päädytään solmun IN1 toiseen alkioon, joka sisältää osoittimen seuraavalla tasolla olevaan solmuun IN3. Tällä tasolla irrotetaan hakuavaimesta seuraavat kaksi bittiä (11), jolloin päädytään ko. solmun neljänteen alkioon, joka osoittaa tietueeseen H.

Osoittimen asemesta sanko voi sisältää (hakuavaimen lisäksi) varsinaisen datatietueen (josta käytetään myös yleisempää nimitystä tietoyksikkö). Näin ollen esim. tilaajaa A koskevat tiedot (kuvio 1) voivat olla sangossa A, tilaajaa B koskevat tiedot sangossa B, jne. Muistin eräässä suoritusmuodossa sankoon on siis talletettu avain-osoitin-pari ja eräässä toisessa suoritusmuodossa avain ja varsinainen data, joskaan avain ei ole välttämätön. Tässä suhteessa kuvataan erilaisia toteutusvaihtoehtoja tarkemmin jäljempänä.

Hakuavain voi olla myös moniulotteinen. Toisin sanoen, hakuavain voi koostua useasta attribuutista (esim. tilaajan sukunimi ja yksi tai useampi etunimi). Tällainen moniulotteinen trie-rakenne on kuvattu esim. kansainvälisessä patenttihakemuksessa PCT/FI95/00319 (julkaisunumero WO 95/34155). Kyseisessä rakenteessa suoritetaan osoitelaskentaa siten, että kustakin dimensiosta valitaan, muista dimensioista riippumatta, kerrallaan tietty ennalta määrätty lukumäärä bittejä. Trie-rakenteen yksittäisen solmun kullekin dimensiolle asetetaan siis kiinteä, muista dimensioista riippumaton raja määräämällä etukäteen kussakin dimensiossa tutkittavien hakuavainbittien lukumäärä. Tällaisella rakenteella saadaan muistipiirien tarve pieneksi silloin, kun hakuavainten arvojen jakaumat ovat etukäteen tiedossa, jolloin rakenne voidaan toteuttaa staattisena.

Mikäli halutaan saada mahdollisuus muokata rakennetta kulloisenkin avainjakauman mukaan tehokkuudeltaan ja muistinkulutukseltaan mahdollisimman optimaaliseksi, on solmujen kokojen muututtava dynaamisesti avainjakauman muuttuessa. Avainjakauman ollessa tasainen voidaan solmukokoja kasvattaa, jotta rakenteesta saadaan matalampi (matalampi rakenne merkitsee nopeampia hakuja). Epätasaisilla avainjakaumilla, joiden yhteydessä muistinkulutus nousee ongelmaksi dynaamisia solmukokoja käyttävissä muistirakenteissa, voidaan puolestaan pitää solmukoot pienenä, jolloin saadaan paikallisesti tasaisempi avainjakauma ja sitä kautta pienempi muistinkulutus. Solmukokojen dynaamiset muutokset edellyttävät osoitelaskennan toteuttamista siten, että digitaalisen trie-rakenteen muodostaman puumaisen hierarkian

kussakin solmussa valitaan solmukohtainen lukumäärä bittejä käytettyjen hakuavaimien muodostamasta bittijonosta. Solmujen dynaaminen muokkaus vaatii luonnollisestikin oman osuutensa prosessointitehosta.

Valinta kiinteän solmukoon ja dynaamisesti muuttuvan solmukoon välillä riippuu mm. siitä, minkälaiseen sovellukseen muisti on tarkoitettu, esim. mikä on tietokantaan tehtävien hakujen, lisäysten ja poistojen määrä ja mitkä ovat kyseisten operaatioiden suhteelliset osuudet.

Muistin tehokkuuteen ja suorituskykyyn vaikuttavat siis mm. trie-rakenteen vaatima muistitila ja trie-rakenteen syvyys. Funktionaalisessa digitaalisessa trie-rakenteessa, jossa on päivitysten yhteydessä kopioitava koko hakupolku juuresta päivityskohtaan asti, muistin suorituskykyyn vaikuttaa lisäksi se, kuinka paljon kopioitavia sanoja tällaisella polulla on. Näin ollen, jos muistissa suoritetaan kompressointia, tulisi sen tapahtua siten, että kopioitavan tiedon lukumäärä ei kompressoinnin seurauksena kasva siinä määrin, että se heikentää muistin suorituskykyä.

Funktionaalisessa trie-rakenteessa on siten ongelmana se, kuinka rakenteessa tulisi suorittaa kompressointia, jotta muistin suorituskyky olisi mahdollisimman hyvä, kun huomioidaan kaikki edellä mainitut tekijät.

Keksinnön yhteenveto

Keksinnön tarkoituksena on saada aikaan ratkaisu edellä kuvattuun ongelmaan. Tämä päämäärä saavutetaan menetelmällä, joka on määritelty itsenäisissä patenttivaatimuksissa. Näistä osa kuvaa rakennetta, jossa käytetään sankoja ja osa rakennetta, jossa ei käytetä sankoja.

Keksinnön ajatuksena on suorittaa neljän alkion kokoisia solmuja (nelisolmuja) sisältävässä trie-rakenteessa kompressointia siten, että nelisolmu ja sen lapsisolmut korvataan ensin yhdellä kokoa 16 olevalla solmulla ja sen jälkeen tämä solmu kompressoidaan siten, että vain nollaosoittimista poikkeavat osoitinarvot talletetaan fyysisesti solmuun. Tässä yhteydessä solmuun talletetaan esim. bittikuvio tai -kartta, joka osoittaa kompressoidussa solmussa käytettävää hakusanaa vastaavan fyysisen muistipaikan solmussa. Bittikuvion avulla kuvataan solmun vakiopituinen alkiotaulukko (16 indeksiä) solmun fyysisten muistipaikkojen taulukoksi.

Menetelmässä suoritetaan siis ensin syvyyssuunnassa peräkkäisille nelisolmuille tasokompressio, jossa nelisolmusta ja sen lapsista muodostetaan yksi uusi solmu, jonka alkiotaulukossa on 16 alkiota. Tämän jälkeen tälle sol-

mulle suoritetaan vielä leveyssuunnassa kompressio poistamalla solmusta nollaosoittimet. Tämä tapahtuu siten, että solmuun talletetaan fyysisesti vain nollasta poikkeavat osoittimet ja nollaosoittimien sijasta em. bittikartta tai -kuvio. Leveyskompressio voidaan suorittaa sinänsä täysin tunnetusti tai tunnettuja periaatteita varioiden.

Keksinnön mukaista ratkaisua käytettäessä pystytään funktionaalisen rakenteen syvyyttä ja muistinkulutusta pienentämään ilman, että päivitysten yhteydessä kopioitavien sanojen lukumäärä pääsee kompression seurauksena oleellisesti kasvamaan ja sitä kautta vähentämään pienentyneen syvyyden ja muistinkulutuksen mukanaan tuomaa suorituskykyhyötyä.

Jotta muistin suorituskyky pysyisi koko ajan optimaalisena, kompressoidulle solmulle on edullista asettaa tietty maksimikapasiteetti, jonka ylittyessä solmu puretaan takaisin nelisolmuksi ja sen lapsisolmuiksi. Näin ollen kompressointia ei myöskään kannata suorittaa, jos nelisolmulla on liikaa lapsia tai lastenlapsia. Kompressointia voidaan suorittaa esim. vain sellaisille joukoille, joissa nelisolmulla on kaksi lapsisolmua tai sellaisille joukoille, joissa nelisolmulla on korkeintaan kahdeksan lastenlasta (jolloin lapsia voi olla myös kolme tai neljä).

Kuvioluettelo

Seuraavassa keksintöä ja sen edullisia toteutustapoja kuvataan tarkemmin viitaten oheisten piirustusten mukaisiin esimerkkeihin, joissa

kuvio 1 havainnollistaa yksiulotteisen digitaalisen trie-rakenteen käyttöä puhelinkeskuksen tilaajatietojen ylläpitämisessä,

kuvio 2 esittää nelisolmujoukkoa, johon sovelletaan keksinnön mukaista menetelmää,

kuvio 3 esittää 16 alkion kokoista solmua, jolla korvataan kuvion 2 mukainen solmujoukko,

kuvio 4 esittää kuvion 3 solmua kompressoituna,

kuvio 5 esittää kuvion 2 nelisolmujoukkoa, kun trie-rakenteessa käytetään lisäksi leveyskompressiota,

kuviot 6 ja 7 esittävät toista kompressointiesimerkkiä,

kuvio 8 havainnollistaa kompressoidun trie-solmun rakennetta, ja

kuvio 9 esittää erästä keksinnön mukaista muistijärjestelyä lohkokaaviotasolla.

Keksinnön yksityiskohtainen kuvaus

Jotta funktionaalisessa trie-rakenteessa pystyttäisiin aikaansaamaan mahdollisimman hyvä yhdistelmä muistihakujen, muistinkulutuksen sekä päivitysten yhteydessä suoritettavien kopiointien kannalta, suoritetaan trie-rakenteessa kompressointia siten, että nelisolmusta ja sen lapsisolmuista muodostetaan yksi kompressoitu solmu korvaamalla ensin nelisolmu ja sen lapsisolmut yhdellä kokoa 16 olevalla solmulla, jolle suoritetaan sen jälkeen kompressointi, jossa solmusta poistetaan kaikki nollaosoittimet. Tämä jälkimmäinen kompressointi voidaan tehdä sinänsä tunnetulla tavalla, kuten jäljempänä kuvataan.

Rakenteessa käytettävän solmukoon valinta nelisolmuksi on perusedellytys sille, että keksinnön mukaisesta menetelmästä saataisiin mahdollisimman suuri hyöty funktionaalisessa rakenteessa. Voidaan nimittäin osoittaa, että n-haarautuvan rakenteen (n-ary trie) trie-solmujen keskimääräisellä muistitilan tarpeella on (tasaisilla avainjakaumilla) minimit arvoilla n=2 ja n=4. Kun näistä arvoista valitaan arvo n=4, saadaan rakenteen syvyys mahdollisimman pieneksi ja lisäksi kompressointi voidaan toteuttaa tehokkaasti (suuressa osassa muistia) siten, että päivitysten yhteydessä kopioitavien sanojen lukumäärä ei pääse kompressoinnin seurauksena liian suureksi verrattuna kompressoimattomaan tilanteeseen.

Keksinnön mukaista menettelyä kuvataan seuraavassa viitaten kuvioiden 2...4 mukaiseen kompressointiesimerkkiin. Kuviossa 2 on esitetty trie-rakenteeseen kuuluva nelisolmu N50, jolla on kaksi lapsisolmua (N51 ja N52). Kummastakin lapsisolmusta on tässä esimerkkitapauksessa kaksi osoitinta alaspäin puussa. Aluksi nelisolmusta ja sen lapsisolmuista muodostetaan tasokompression avulla kuvion 3 mukainen solmu N60, jonka loogisessa alkiotaulukossa on 16 alkiota. Kuviossa 2 on merkitty solmun N50 alkiotaulukkoindeksit niitä vastaavien alkioiden yläpuolelle ja kuvioon 3 on merkitty kyseisiä indeksejä vastaavat neljän alkion ryhmät. Lapsisolmujen osoittimet A...D asettuvat kuvion 3 mukaisille paikoille (loogisia indeksejä 0, 1, 12 ja 14 vastaavissa paikoissa).

Tasokompression jälkeen solmulle N60 suoritetaan leveyskompressio tallettamalla muistiin vain ne osoittimet, jotka ovat erisuuria kuin nollaosoitin. Nollaosoittimesta poikkeavien osoittimien lisäksi trie-solmun yhteyteen talletetaan bittikuvio tai -kartta, jonka perusteella pystytään määrittämään, onko solmun alkiotaulukon loogista indeksiä vastaava osoitin nollaosoitin vai ei, ja

jos ei ole, missä ko. loogista indeksiä vastaava osoitin sijaitsee fyysisesti solmussa. Kompressointia käytettäessä solmun vakiopituinen (16 kpl) alkiotaulukko kuvautuu bittikuvion avulla fyysisten muistipaikkojen taulukoksi, jonka pituus vaihtelee sen mukaan, kuinka monta nollaosoitinta solmussa kulloinkin on. Huomattakoon siis, että leveyskompression yhteydessä solmun looginen koko (eli alkiotaulukon koko) ei muutu, mutta sen sijaan solmun fyysinen koko pienenee, koska kompressoidussa solmussa nollaosoittimet eivät vaadi muistitilaa.

Keksinnön erään toteutustavan mukaisesti kaikki nollaosoittimesta poikkeavat osoittimet talletetaan solmuun peräkkäin siten, että niiden keskinäinen järjestys säilyy samana kuin alkiotaulukossa ja erillisen bittikuvion avulla kerrotaan kunkin talletetun osoittimen fyysinen indeksi (fyysinen sijainti).

Tällä tavoin saadaan kuvion 4 mukainen kompressoitu solmu N70, jossa kaikki nollaosoittimesta poikkeavat osoittimet ovat peräkkäin. Solmussa on vain neljä fyysistä alkiota (osoittimet A...D) ja niiden lisäksi solmuun talletetaan bittikuvio BP1, joka indikoi hakuavaimesta muodostettua alkiotaulukkoindeksiä vastaavan osoittimen fyysisen sijainnin solmussa. Tässä tapauksessa bittikuviossa on yksi bitti alkiotaulukon jokaista alkiota (loogista indeksiä) kohti ja kukin bitti kertoo, onko sitä vastaavassa alkiossa nollaosoittimesta poikkeava osoitin vai nollaosoitin. Kuvion esimerkkitapauksessa on ykkösellä osoitettu nollaosoittimesta poikkeavaa osoitinta ja nollalla nollaosoitinta. Koska osoittimet talletetaan kompressoituun solmuun järjestys säilyttäen (eikä nollaosoittimille varata tilaa), tiedetään kuvion 4 mukaisessa kompressoidussa solmussa, että alkiotaulukkoindeksiä 0 vastaa nollasta poikkeava osoitin, jolloin sen fyysinen indeksi on nolla, että alkiotaulukkoindeksiä 1 vastaa nollasta poikkeava osoitin, jolloin sen fyysinen indeksi on yksi, alkiotaulukkoindeksejä 2...11 vastaavat nollaosoittimet, alkiotaulukkotaulukkoindeksiä 12 vastaa nollasta poikkeava osoitin, jolloin sen fyysinen indeksi on kaksi, alkiotaulukkoindeksiä 13 vastaa nollaosoitin, alkiotaulukkoindeksiä 14 vastaa nollasta poikkeava osoitin, jolloin sen fyysinen indeksi on kolme, ja alkiotaulukkoindeksiä 15 vastaa nollaosoitin. Näin ollen solmusta löydetään helposti se osoitin, joka vastaa sitä loogista indeksiä, joka muodostetaan hakuavainbiteistä. Tällainen kompressointitapa, jossa kompressoidun solmun bittikuviossa on yksi bitti alkiotaulukon jokaista alkiota (loogista indeksiä) kohti on sinänsä tunnettu. Tällaiseen ratkaisuun viitataan esim. US-patentissa 5,276,868.

Osoitelaskenta tapahtuu kompressoidun solmun kohdalla siten, että hakuavaimesta irrotetuista biteistä (4 kpl) muodostetaan ensin normaalilla tavalla (looginen) alkiotaulukkoindeksi. Tämän jälkeen luetaan bittikuviosta tätä indeksiä vastaava bitti. Mikäli bitti osoittaa, että kysymyksessä on nollaosoitin, päättyy haku epäonnistuneena. Mikäli bitti osoittaa, että kysymyksessä on nollasta poikkeava osoitin, määritetään bittikuvion perusteella kyseistä alkiotaulukkoindeksiä vastaavan osoittimen fyysinen sijainti (fyysinen indeksi). Käytettäessä kuvion 4 kaltaista bittikuviota, jossa on yksi bitti alkiotaulukon jokaista indeksiä kohti fyysinen indeksi saadaan suoraan laskemalla ykkösbittien lukumäärä bittikuvion alusta alkiotaulukkoindeksiä vastaavaan bittiin asti.

Tässä toteutustavassa bittikuviossa on siis 16 kappaletta bittejä ja (fyysisessä) solmussa on (16-P0) kappaletta osoittimia, kun P0 on solmun (loogisessa) alkiotaulukossa olevien nollaosoittimien lukumäärä. Bittikuviossa käytettävien bittien lukumäärä voi kuitenkin poiketa loogisten indeksien lukumäärästä ja bittikuviossa voidaan käyttää erilaisia koodauksia, joiden avulla selvitetään loogista indeksiä vastaava fyysinen muistipaikka. Edellä esitetyllä ratkaisulla saavutetaan kuitenkin lyhyt bittikuvio ja yksinkertainen koodausmekanismi.

Trie-rakenteessa voidaan käyttää edellä kuvatun kompressointitavan lisäksi myös leveyskompressiota, joka kohdistetaan yksittäisiin solmuihin. Tällöin siis nelisolmussa on (4-P0) kappaletta osoittimia ja esim. neljän bitin pituinen bittikuvio, jonka kukin bitti kertoo, onko sitä vastaavassa alkiossa nollaosoitin vai nollasta poikkeava osoitin. Kuviossa 5 on esitetty kuviota 2 vastaava tilanne, kun kaikki nelisolmut ovat leveyskompressoituja nelisolmuja. Solmuja on tässä tapauksessa merkitty viitemerkeillä N80...N82 ja kunkin solmun nelibittistä bittikuviota viitemerkillä BP2. Kuvion 5 mukaisesta solmuryhmästä muodostetaan kuvion 4 mukainen kompressoitu solmu N70. Nelisolmut, jotka kompressoidaan keksinnön mukaisella menetelmällä voivat siis olla tavallisia nelisolmuja tai leveyskompressoituja nelisolmuja, jotka on kompressoitu esim. kuvion 5 esittämällä tavalla. Kompressoitavassa ryhmässä voi myös olla molemman tyyppisiä solmuja. (Termi nelisolmu viittaa yleisesti solmuun, jonka loogisessa alkiotaulukossa on neljä alkiota.)

Käytännössä vaatii jokainen edellä esitetyn esimerkin solmuista yhteensä 5 sanaa (solmut N50...N52) tai 3 sanaa (solmut N80...N82) muistitilaa (sana jokaiselle osoittimelle ja lisäksi yksi solmun tyyppi-informaation sisältävä sana). Kuvion 5 mukaisen yhdistelmän tilantarve on siis 9 sanaa. Kompressoi-

dun solmun N70 tilantarve on puolestaan 5 sanaa (jos bittikuvio mahtuu tyyppi-informaation sisältävään sanaan). Jos esim. solmuun N82 lisätään osoitin, joudutaan kuvion 8 esimerkkitapauksessa kopioimaan yhteensä 6 sanaa (solmut N80 ja N82). Kun solmut on korvattu kompressoidulla solmulla N70, joudutaan kopioimaan 5 sanaa. Kompressointi on siis tässä esimerkkitapauksessa suoritettu siten, että päivityksen yhteydessä kopioitavien sanojen lukumäärä on pienentynyt yhdellä. Lisäksi rakenteen vaatima muistitila ja tasojen lukumäärä on pienentynyt. Kuvion 2 mukaiseen lähtökohtatilanteeseen verrattuna on saavutettu hyöty vieläkin suurempi.

Kuvioissa 6 ja 7 on esitetty toinen esimerkkitapaus, jossa kuvion 6 (tavallisella) nelisolmulla N90 on kolme (tavallista) lapsisolmua, joista on yhteensä 6 osoitinta (A...F) alaspäin. Kun tämä yhdistelmä kompressoidaan edellä kuvatulla tavalla, saadaan kuvion 7 mukainen kompressoitu solmu N100.

Solmun tyyppi-informaation lisäksi kussakin tavallisessa nelisolmussa on oltava ainoastaan alkiotaulukko (osoittimet). Mikäli ei käytetä muita kompressiotapoja, tyyppi-informaatio kertoo sen, onko kysymyksessä tavallinen nelisolmu vai keksinnön mukaisesti kompressoitu solmu, jolloin tiedetään, onko tutkittavien bittien lukumäärä kaksi vai neljä. Mikäli käytetään lisäksi muita kompressointitapoja (leveys- ja/tai polkukompressiota), tyyppi-informaatio kertoo vastaavasti, mikä solmutyyppi on kysymyksessä. Kompressoidussa solmussa on lisäksi aina em. bittikuvio, joka voi olla myös kompressoimattomissa solmuissa, erityisesti, jos käytännön tilanne on sellainen, että solmussa on ylimääräisiä (käyttämättömiä) bittejä, joiden paikalle bittikuvio voidaan laittaa ilman, että solmun vaatima muistitila kasvaa.

Kuviossa 8 on havainnollistettu kompressoidun nelisolmun rakennetta. Minimikokoonpanossaan kompressoitu solmu käsittää siis kolme osaa: tyyppi-informaatiokentän, joka kertoo solmun tyypin (viitenumero 111), bittikuvion sisältävän kentän 112 ja alkiotaulukon (viitenumero 113), jossa olevien alkioiden (osoittimien) lukumäärä on edellä esitetyn mukainen.

Solmun tyyppi-informaatio voidaan tallettaa myös sen isäsolmun osoittimeen, jos siinä on ylimääräisiä bittejä tähän tarkoitukseen. Esim. 32 bittisessä arkkitehtuurissa osoittimen kahdella "alimmalla" bitillä voidaan koodata osoittimeen tieto esim. siitä, onko kysymyksessä nollaosoitin (tyhjä alkio) vai osoittaako osoitin tavalliseen trie-solmuun, sankoon vai kompressoituun trie-solmuun. Sangottoman rakenteen tapauksessa talletetaan muistiin tieto

siitä, osoittaako osoitin kompressoimattomaan solmuun, kompressoituun solmuun vai tietoyksikköön.

Kuten alussa jo mainittiin, edellä kuvattu kompressointiperiaate koskee myös sangotonta trie-rakennetta. Sangottomassa rakenteessa saavutetaan keksinnön mukaisella ratkaisulla jopa suurempi hyöty kuin sangollisessa rakenteessa. Tämä johtuu siitä, että solmujen täyttöasteet ovat yleensä suurempia trie-puun yläpäässä kuin trie-puun alapäässä. Sankojen avulla voidaan yhdistellä trie-puun alapäässä olevia solmuja, joten sangottomassa rakenteessa on siis yleensä enemmän solmuja, joissa tyhjien alkioiden lukumäärä on suurempi kuin trien yläpäässä. (Sangottomassa rakenteessa sankoa vastaa lehtisolmu, joka sisältää tyypillisesti osoittimen tietoyksikköön tai tietoyksikön.)

Jotta päivitysten yhteydessä kopioitavien sanojen lukumäärää ei kasvatettaisi liian isoksi ja sillä tavoin heikennettäisi kompressiolla saavutettavaa hyötyä, kompressoidulle solmulle on edullista asettaa jokin ennalta määrätty maksimikapasiteetti (maksimilukumäärä osoittimia). Tämä lukumäärä voi olla esim. kahdeksan tai kymmenen. Jos rakenteeseen tehdään lisäyksiä niin, että kompressoidun solmun osoittimien lukumäärä kasvaa valitun maksimikapasiteetin yli, puretaan kompressoitu solmu takaisin kahdella eri tasolla oleviksi nelisolmuiksi (kuviot 2, 5 ja 6). Purkuoperaatio on käänteinen kompressoinnille.

Edellä esitetystä johtuen kompressointimenetelmää voidaan soveltaa vain sellaisille solmuryhmille, jotka täyttävät ennalta määrätyn kompressointiehdon, joka on asetettu siten, että menetelmästä saatava hyöty on mahdollisimman suuri. Kompressointi voidaan suorittaa esim. vain sellaisille solmuryhmille, joissa nelisolmulla on kaksi lasta (jolloin lastenlapsia on korkeintaan kahdeksan), joista vähintään toisen on oltava tavallinen trie-solmu (vrt. kuvio 2) tai sangottoman rakenteen tapauksessa sisäsolmu. Vaihtoehtoisesti menetelmää voidaan soveltaa kaikille sellaisille solmuryhmille, joissa nelisolmulla on korkeintaan kahdeksan lastenlasta (vrt. kuvio 6). Ensimmäisen vaihtoehdon etuna on se, että lastenlapsien lukumäärää ei tarvitse tutkia, kun taas toisen vaihtoehdon etuna on se, että sitä voidaan soveltaa laajemmin eli myös sellaisille nelisolmuille, joilla on kolme tai neljä lasta. Eräs tapa on määrätä osoittimien lukumäärien rajat niin, että kompressointiraja on pienempi kuin kompression purkamisraja, esim. niin, että kompressointi suoritetaan nelisolmuille, joilla on korkeintaan kahdeksan lastenlasta, mutta kompressio puretaan vasta, kun kompressoidun solmun osoittimien lukumäärä ylittää kymmenen. Tällä tavoin

saadaan aikaan se, että purkua ei koskaan tarvitse suorittaa kompressoinnin jälkeen heti seuraavan päivityksen yhteydessä.

Keksinnön mukaiseen muistiin on edullista yhdistää myös edellä lyhyesti viitattu polkukompressio. Polkukompressio voidaan toteuttaa jollakin sinänsä tunnetulla tavalla, esim. siten kuin kuvataan hakijan aikaisemmassa patenttihakemuksessa PCT/FI98/00191. Tässä julkaisussa kuvatussa menetelmässä polkukompressio suoritetaan korvaamalla keskenään peräkkäisten nelisolmujen joukkoja kompressoiduilla solmuilla siten, että yksittäinen joukko, joka muodostuu keskenään peräkkäisistä nelisolmuista, joista jokaisesta on ainoastaan yksi osoite alemman tason nelisolmuun korvataan kompressoidulla solmulla, johon talletetaan osoite siihen nelisolmuun, johon korvattavassa joukossa alimpana oleva solmu osoittaa. Lisäksi solmuun talletetaan tieto sen hakusanan arvosta, jolla mainittu osoite löytyy sekä tieto niiden bittien kokonaislukumäärästä, joista korvattavassa joukossa muodostetaan hakusanat. Jos trie-rakenteessa käytetään polkukompressiota, on vähintään toisen lapsisolmuista oltava polkukompressoimaton sisäsolmu, jotta keksinnön mukainen kompressio voitaisiin tehdä. Koska polkukompressio voidaan toteuttaa sinänsä tunnetuilla tavoilla, sitä ei kuvata tässä yhteydessä tarkemmin.

Yksiulotteisesta hakurakenteesta saadaan moniulotteinen (yleisesti ottaen n-ulotteinen) sinänsä tunnetulla tavalla bittilimitystä käyttäen. Bittilimitystä on kuvattu esim. edellä mainitussa kansainvälisessä patenttihakemuksessa PCT/FI98/00191, josta kiinnostunut lukija löytää halutessaan keksintöön liittyvää taustatietoa.

Kuviossa 9 on esitetty erästä keksinnön mukaista muistia lohkokaaviotasolla. Kutakin dimensiota varten on oma tulorekisteri, joten tulorekisterejä on siis yhteensä n kappaletta. Näihin tulorekistereihin, joita on merkitty viitemerkeillä $R_1...R_n$, talletetaan kunkin dimension hakuavain, kukin omaan rekisteriinsä. Tulorekisterit on kytketty rekisteriin TR, johon muodostetaan edellä kuvattu hakusana käytetyn bittilimitysmenetelmän mukaisesti. Rekisteri TR on kytketty summaimen S kautta muistin MEM osoitesisäänmenoon. Muistin ulostulo on puolestaan kytketty osoiterekisterille AR, jonka ulostulo on puolestaan kytketty summaimelle S. Aluksi luetaan jokaisesta rekisteristä valitut bitit oikeaan järjestykseen yhteisrekisteriin TR. Osoiterekisteriin AR on aluksi talletettu ensimmäisen trie-solmun aloitusosoite, jolloin tähän osoitteeseen summataan summaimessa S se osoite, joka saadaan offset-osoitteena rekisteriltä TR. Tämä osoite syötetään muistin MEM osoitesisäänmenoon, jolloin muistin

dataulostulosta saadaan seuraavan trie-solmun aloitusosoite, joka kirjoitetaan osoiterekisteriin AR siellä olleen edellisen osoitteen päälle. Tämän jälkeen ladataan tulorekistereistä jälleen seuraavaksi valittavat bitit oikeaan järjestykseen yhteisrekisteriin TR ja näin saatu taulukko-osoite summataan ko. taulukon (eli trie-solmun) aloitusosoitteeseen, joka saadaan osoiterekisteristä AR. Tämä osoite syötetään jälleen muistin MEM osoitesisäänmenoon, jolloin muistin dataulostulosta saadaan seuraavan solmun aloitusosoite. Edellä kuvattuja menettelyä toistetaan kunnes on edetty haluttuun pisteeseen asti ja voidaan suorittaa talletus tai lukea haluttu tietue.

Ohjauslogiikka CL huolehtii siitä, että kompressoidun solmun kohdalla saadaan dataulostulosta oikean solmun osoite. Kun kyseessä on kompressoitu solmu, ohjauslogiikka lukee alkiotaulukkoindeksiä vastaavan bitin solmun bittikuviosta. Mikäli ko. bitti osoittaa, että kysymyksessä on nollaosoittimesta poikkeava osoitin, ohjauslogiikka määrittää bittikuvion perusteella fyysisen indeksin ja tallettaa sen rekisteriin TR loogisen indeksin tilalle. Tällöin fyysinen indeksi summataan (loogisen indeksin sijasta) summaimessa S rekisteristä AR saatavaan solmun aloitusosoitteeseen. Ohjauslogiikka huolehtii myös solmujen kompressoinnista sekä siitä, että kussakin solmussa otetaan rekistereistä oikea määrä bittejä (2 tai 4 kpl).

Osoitelaskennan nopeuteen voidaan vaikuttaa sillä, minkälainen laitetoteutus valitaan. Koska eteneminen tapahtuu edellä esitettyjen bittimanipulaatioiden avulla, voidaan osoitelaskentaa nopeuttaa siirtymällä yhden prosessorin käytöstä moniprosessoriympäristöön, jossa suoritetaan rinnakkaista prosessointia. Moniprosessoriympäristölle vaihtoehtoinen toteutustapa on ASIC-piiri.

Keksinnön mukaisen ratkaisun ansiosta säästetään jokaisen nelisolmuryhmän kohdalla tyypillisesti muutamaa tietokoneen sanan pituutta vastaava määrä muistitilaa (sanassa on tyypillisesti 32 bittiä). Bittikuviota varten ei käytännössä yleensä tarvita lainkaan lisätilaa, koska solmussa on käytännössä sellaisia tietokenttiä, jotka eivät tarvitse kokonaista sanan pituutta (muistia varataan sanan pituus kerrallaan), esim. tyyppi-informaation sisältävä kenttä. Lisäksi tasojen lukumäärä pienenee paikallisesti yhdellä ja parhaimmassa tapauksessa koko trie-puun syvyys pienenee puoleen. Samalla voidaan myös saada päivitysten yhteydessä kopioitavien sanojen lukumäärä pienemmäksi.

Vaikka keksintöä on edellä selostettu viitaten oheisten piirustusten mukaisiin esimerkkeihin, on selvää, ettei keksintö ole rajoittunut siihen, vaan sitä

voidaan muunnella oheisissa patenttivaatimuksissa esitetyn keksinnöllisen ajatuksen puitteissa. Kompressointi voidaan toteuttaa esim. vain osassa muistia, esim. vain sisäsolmuille. Osoitelaskenta voi myös jatkua vielä sangossakin, edellyttäen, että bittejä on vielä tutkimatta. Alussa esitettyä sangon määritelmää onkin laajennettava siten, että sanko on tietorakenne, joka voi sisältää myös toisen trie-rakenteen. Useita keksinnön mukaisia hakemistorakenteita voidaan siis linkittää peräkkäin siten, että sankoon on talletettu toinen hakemistorakenne (eli toinen trie-rakenne) tai sangon sisältämä osoitin osoittaa toiseen hakemistorakenteeseen. Viittaus sangosta tapahtuu suoraan seuraavan hakemistorakenteen juurisolmuun. Myös lehtisolmusta voi olla viittaus seuraavan hakemistorakenteen juurisolmuun. Yleisesti ottaen voidaan todeta, että sanko sisältää ainakin yhden elementin siten, että yksittäisen elementin tyyppi on valittu joukosta, joka käsittää tietoyksikön, osoittimen talletettuun tietoyksikköön, osoittimen toiseen hakemistorakenteeseen ja toisen hakemistorakenteen. Sankojen tarkempi toteutus riippuu sovelluksesta. Useimmiten kaikki sangoissa olevat elementit ovat samaa tyyppiä, joka on joko tietoyksikkö tai osoitin tietoyksikköön. Lehden tai sangon eri elementtityyppeihin voi myös liittyä lisätietoa, esim. tyyppiä "osoitin tietoyksikköön" tai tyyppiä "tietoyksikkö" oleviin elementteihin voi liittyä hakuavain, kuten edellä mainittiin. Sovelluksissa, joissa muistiin talletetaan merkkijonoja, sangossa voi myös olla elementtipareja, esim. siten, että sangon kaikki parit ovat joko osoitin tietoyksikköön/osoitin hakemistorakenteeseen -pareja tai tietoyksikkö/osoitin hakemistorakenteeseen -pareja tai tietoyksikkö/hakemistorakenne -pareja. Tällöin voidaan esim. merkkijonon etuosa (prefix) tallettaa tietoyksikköön ja jatkaa hakua tietoyksikköä vastaavaan pariin kuuluvasta hakemistorakenteesta.

Patenttivaatimukset

1. Menetelmä funktionaalisen muistin toteuttamiseksi, johon muistiin tieto talletetaan tietoyksikköinä, joista jokaiselle varataan muistissa oma muistitilansa, jonka menetelmän mukaisesti

- muisti toteutetaan hakemistorakenteena, joka muodostuu puumaisesta hierarkiasta, jossa on useilla eri tasoilla olevia solmuja, jolloin yksittäinen solmu voi olla (i) trie-solmu, johon liittyy looginen taulukko, jonka yksittäinen alkio voi sisältää puumaisessa hierarkiassa alempana olevaan solmuun osoittavan osoittimen ja jonka yksittäinen alkio voi myös olla tyhjä, jolloin alkion sisältö vastaa nollaosoitinta, jossa taulukossa olevien alkioiden lukumäärä vastaa jotakin kakkosen potenssia, tai (ii) sanko, joka sisältää ainakin yhden elementin siten, että sangon yksittäisen elementin tyyppi on valittu joukosta, joka käsittää tietoyksikön, osoittimen talletettuun tietoyksikköön, osoittimen toiseen hakemistorakenteeseen ja toisen hakemistorakenteen,

- hakemistorakenteessa suoritetaan osoitelaskentaa siten, että

- (a) valitaan puumaisessa hierarkiassa ylimmällä tasolla olevassa solmussa tietty lukumäärä bittejä käytettyjen hakuavaimien muodostamasta bittijonosta, muodostetaan valituista biteistä hakusana, jonka perusteella haetaan kyseisessä solmussa seuraavan solmun osoite ja edetään mainittuun solmuun,

- (b) valitaan käytettyjen hakuavaimien muodostaman bittijonon vielä valitsematta olevien bittien joukosta ennalta määrätty lukumäärä bittejä ja muodostetaan valituista biteistä hakusana, jonka avulla haetaan jälleen uuden, alemmalla tasolla olevan solmun osoite sen solmun taulukosta, johon on edetty,

- toistetaan askelta (b) kunnes päästään nollaosoittimen sisältävään alkioon tai kunnes uuden, alemmalla tasolla olevan solmun osoite on sangon osoite,

jolloin ne solmut, joihin tietystä solmusta on osoittimet ovat mainitun tietyn solmun lapsia ja ne solmut, joihin lapsista on osoittimet ovat mainitun tietyn solmun lastenlapsia,

t u n n e t t u siitä, että

trie-solmut toteutetaan neljän alkion kokoisina nelisolmuina ja ainakin osassa hakemistorakennetta korvataan keskenään peräkkäisten solmujen joukkoja kompressoiduilla solmuilla siten, että

(a) yksittäinen joukko, joka muodostuu tietystä nelisolmusta ja sen lapsista korvataan solmulla, jonka loogisessa taulukossa on 16 alkiota, ja

(b) mainitusta 16 alkion solmusta muodostetaan sinänsä tunnettu kompressoitu solmu tallettamalla solmuun fyysisesti vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio, jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

2. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että korvausta suoritetaan hakemistorakenteessa kaikille sellaisille joukoille, joissa nelisolmulla on kaksi lasta.

3. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että korvausta suoritetaan hakemistorakenteessa kaikille sellaisille joukoille, joissa nelisolmulla on korkeintaan kahdeksan lastenlasta.

4. Patenttivaatimuksen 2 tai 3 mukainen menetelmä, t u n n e t t u siitä, että kompressoidun solmun osoittimien lukumäärälle asetetaan yläraja, jolloin rajan ylittyessä kompressoitu solmu puretaan takaisin nelisolmuksi ja lapsisolmuiksi.

5. Patenttivaatimuksen 4 mukainen menetelmä, t u n n e t t u siitä, että ylärajana käytetään kahdeksaa osoitinta.

6. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että ylärajana käytetään kymmentä osoitinta.

7. Patenttivaatimuksen 2 tai 3 mukainen menetelmä, t u n n e t t u siitä, että ainakin osalle rakenteen nelisolmuista (N80...N82) suoritetaan lisäksi kompressointi siten, että solmuun talletetaan fyysisesti vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio (BP2), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

8. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että nollaosoittimista poikkeavat osoittimet talletetaan kompressoituun solmuun peräkkäin samassa järjestyksessä, joka niillä on mainitussa taulukossa.

9. Patenttivaatimuksen 8 mukainen menetelmä, t u n n e t t u siitä, että bittikuviossa on yksi bitti taulukon jokaista alkiota kohti, jolloin kukin bitti indikoi, onko sitä vastaavassa alkiossa nollaosoitin vai nollasta poikkeava osoitin.

10. Patenttivaatimuksen 8 mukainen menetelmä, t u n n e t t u siitä, että bittikuviota varten on varattu tila hakemistorakenteen kaikkiin trie-solmuihin.

11. Patenttivaatimuksen 8 mukainen menetelmä, t u n n e t t u siitä, että bittikuviota bittikuviota varten on varattu tila vain kompressoituihin solmuihin.

12. Menetelmä funktionaalisen muistin toteuttamiseksi, johon muistiin tieto talletetaan tietoyksikköinä, joista jokaiselle varataan muistissa oma muistitilansa, jonka menetelmän mukaisesti

- muisti toteutetaan hakemistorakenteena, joka muodostuu puumaisesta hierarkiasta, jossa on useilla eri hierarkiatasoilla olevia solmuja, jolloin yksittäinen solmu voi olla (i) sisäsolmu, johon liittyy looginen taulukko, jonka yksittäinen alkio voi sisältää puumaisessa hierarkiassa alempana olevaan solmuun osoittavan osoittimen ja jonka yksittäinen alkio myös olla tyhjä, jolloin alkion sisältö vastaa nollaosoitinta, jossa taulukossa olevien alkioiden lukumäärä vastaa jotakin kakkosen potenssia, tai (ii) lehti, joka sisältää elementin, jonka tyyppi on valittu joukosta, joka käsittää osoittimen talletettuun tietoyksikköön, tietoyksikön ja osoittimen toisen hakemistorakenteen solmuun,

- hakemistorakenteessa suoritetaan osoitelaskentaa siten, että

- (a) valitaan puumaisessa hierarkiassa ylimmällä tasolla olevassa solmussa tietty lukumäärä bittejä käytettyjen hakuavaimien muodostamasta bittijonosta, muodostetaan valituista biteistä hakusana, jonka perusteella haetaan kyseisessä solmussa seuraavan solmun osoite ja edetään mainittuun solmuun,

- (b) valitaan käytettyjen hakuavaimien muodostaman bittijonon vielä valitsematta olevien bittien joukosta tietty lukumäärä bittejä ja muodostetaan valituista biteistä hakusana, jonka avulla haetaan jälleen uuden, alemmalla tasolla olevan solmun osoite sen solmun taulukosta, johon on edetty,

- toistetaan askelta (b) kunnes päästään tyhjään alkioon tai kunnes uuden, alemmalla tasolla olevan solmun osoite on lehden osoite,

jolloin ne solmut, joihin tietystä solmusta on osoittimet ovat mainitun tietyn solmun lapsia ja ne solmut, joihin lapsista on osoittimet ovat mainitun tietyn solmun lastenlapsia,

t u n n e t t u siitä, että

sisäsolmut toteutetaan neljän alkion kokoisina nelisolmuina ja ainakin osassa hakemistorakennetta korvataan keskenään peräkkäisten solmujen joukkoja kompressoiduilla solmuilla siten, että

- yksittäinen joukko, joka muodostuu tietystä nelisolmusta ja sen lapsista korvataan solmulla, jonka loogisessa taulukossa on 16 alkiota, ja

- mainitusta 16 alkion solmusta muodostetaan sinänsä tunnettu kompressoitu solmu tallettamalla solmuun fyysisesti vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio, jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

13. Patenttivaatimuksen 12 mukainen menetelmä, t u n n e t t u siitä, että korvausta suoritetaan hakemistorakenteessa kaikille sellaisille joukoille, joissa nelisolmulla on kaksi lasta.

14. Patenttivaatimuksen 12 mukainen menetelmä, t u n n e t t u siitä, että korvausta suoritetaan hakemistorakenteessa kaikille sellaisille joukoille, joissa nelisolmulla on korkeintaan kahdeksan lastenlasta.

15. Patenttivaatimuksen 13 tai 14 mukainen menetelmä, t u n n e t t u siitä, että kompressoidun solmun osoittimien lukumäärälle asetetaan yläraja, jolloin rajan ylittyessä kompressoitu solmu puretaan takaisin nelisolmuksi ja lapsisolmuiksi.

16. Patenttivaatimuksen 15 mukainen menetelmä, t u n n e t t u siitä, että ylärajana käytetään kahdeksaa osoitinta.

17. Patenttivaatimuksen 15 mukainen menetelmä, t u n n e t t u siitä, että ylärajana käytetään kymmentä osoitinta.

18. Patenttivaatimuksen 13 tai 14 mukainen menetelmä, t u n n e t t u siitä, että ainakin osalle rakenteen nelisolmuista suoritetaan lisäksi kompressointi siten, että solmuun talletetaan fyysisesti vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio (BP2), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

19. Patenttivaatimuksen 12 mukainen menetelmä, t u n n e t t u siitä, että nollaosoittimista poikkeavat osoittimet talletetaan kompressoituun solmuun peräkkäin samassa järjestyksessä, joka niillä on mainitussa taulukossa.

20. Patenttivaatimuksen 19 mukainen menetelmä, t u n n e t t u siitä, että bittikuviossa on yksi bitti taulukon jokaista alkiota kohti, jolloin kukin bitti indikoi, onko sitä vastaavassa alkiossa nollaosoitin vai nollasta poikkeava osoitin.

21. Patenttivaatimuksen 19 mukainen menetelmä, t u n n e t t u siitä, että bittikuviota varten on varattu tila hakemistorakenteen kaikkiin trie-solmuihin.

22. Patenttivaatimuksen 19 mukainen menetelmä, t u n n e t t u siitä, että bittikuviota bittikuviota varten on varattu tila vain kompressoituihin solmuihin.

23. Muistijärjestely tietoyksikköjen tallettamiseksi, joka muistijärjestely käsittää hakemistorakenteen, jossa edetään käyttämällä hakusanoja, jotka muodostetaan kulloinkin käytettyjen hakuavaimien muodostamasta bittijonosta, joka hakemistorakenne muodostuu puumaisesta hierarkiasta, jossa on useilla eri hierarkiatasoilla olevia solmuja, jolloin yksittäinen solmu voi olla (i) trie-solmu, johon liittyy looginen taulukko, jonka yksittäinen alkio voi sisältää puumaisessa hierarkiassa alempana olevaan solmuun osoittavan osoittimen ja jonka yksittäinen alkio voi myös olla tyhjä, jolloin alkion sisältö vastaa nollaosoitinta, jossa taulukossa olevien alkioiden lukumäärä vastaa jotakin kakkosen potenssia, tai (ii) sanko, joka sisältää ainakin yhden elementin siten, että sangon yksittäisen elementin tyyppi on valittu joukosta, joka käsittää tietoyksikön, osoittimen talletettuun tietoyksikköön, osoittimen toisen hakemistorakenteen solmuun ja toisen hakemistorakenteen,

t u n n e t t u siitä, että

osa trie-solmuista on nelisolmuja, joiden loogisessa taulukossa on neljä alkiota ja osa solmuja, joiden loogisessa taulukossa on 16 alkiota ja joihin on fyysisesti talletettu vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio (BP1), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

24. Patenttivaatimuksen 23 mukainen menetelmä, t u n n e t t u siitä, että ainakin osaan mainituista nelisolmuista on fyysisesti talletettu osoittimista vain ne, jotka poikkeavat nollaosoittimista ja niiden lisäksi bittikuvio (BP2), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

25. Muistijärjestely tietoyksikköjen tallettamiseksi, joka muistijärjestely käsittää hakemistorakenteen, jossa edetään käyttämällä hakusanoja, jotka muodostetaan kulloinkin käytettyjen hakuavaimien muodostamasta bittijonosta, joka hakemistorakenne muodostuu puumaisesta hierarkiasta, jossa on useilla eri hierarkiatasoilla olevia solmuja, jolloin yksittäinen solmu voi olla (i) sisäsolmu, johon liittyy looginen taulukko, jonka yksittäinen alkio voi sisältää puumaisessa hierarkiassa alempana olevaan solmuun osoittavan osoittimen ja jonka yksittäinen alkio voi myös olla tyhjä, jolloin alkion sisältö vastaa nollaosoitinta, jossa taulukossa olevien alkioiden lukumäärä vastaa jotakin kakkosen potens-

sia, tai (ii) lehti, joka sisältää ainakin yhden elementin, jonka tyyppi on yksi joukosta, joka käsittää osoittimen talletettuun tietoyksikköön ja osoittimen toisen hakemistorakenteen solmuun,

t u n n e t t u siitä, että

osa sisäsolmuista on nelisolmuja, joiden loogisessa taulukossa on neljä alkiota ja osa solmuja, joiden loogisessa taulukossa on 16 alkiota ja joihin on fyysisesti talletettu vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio (BP1), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

26. Patenttivaatimuksen 23 mukainen menetelmä, t u n n e t t u siitä, että ainakin osaan mainituista nelisolmuista on fyysisesti talletettu osoittimista vain ne, jotka poikkeavat nollaosoittimista ja niiden lisäksi bittikuvio (BP2), jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa.

(57) Tiivistelmä

Keksintö koskee menetelmää funktionaalisen muistin toteuttamiseksi sekä muistijärjestelyä. Muisti toteutetaan hakemistorakenteena, joka muodostuu puumaisesta hierarkiasta, jossa on useilla eri hierarkiatasoilla olevia solmuja, jolloin yksittäinen solmu voi olla (i) trie-solmu, johon liittyy looginen taulukko, jonka yksittäinen alkio voi sisältää hierarkiassa alempana olevaan solmuun osoittavan osoittimen, tai (ii) sanko, joka sisältää ainakin yhden elementin siten, että sangon yksittäisen elementin tyyppi on valittu joukosta, joka käsittää mm. tietoyksikön tai osoittimen talletettuun tietoyksikköön. Jotta funktionaalisen trie-rakenteen suorituskyky saataisiin mahdollisimman hyväksi, trie-solmut toteutetaan neljän alkion kokoisina nelisolmuina ja ainakin osassa hakemistorakennetta korvataan keskenään peräkkäisten nelisolmujen joukkoja kompressoiduilla solmuilla siten, että (a) yksittäinen joukko, joka muodostuu tietystä nelisolmusta ja sen lapsista korvataan solmulla, jonka loogisessa taulukossa on 16 alkiota, ja (b) mainitusta 16 alkion solmusta muodostetaan sinänsä tunnettu kompressoitu solmu tallettamalla solmuun fyysisesti vain nollaosoittimista poikkeavat osoittimet ja niiden lisäksi bittikuvio, jonka perusteella voidaan määrittää hakusanaa vastaava fyysinen muistipaikka solmussa. Keksintö koskee myöskin rakennetta, jossa ei käytetä sankoja.

(kuvio 3)

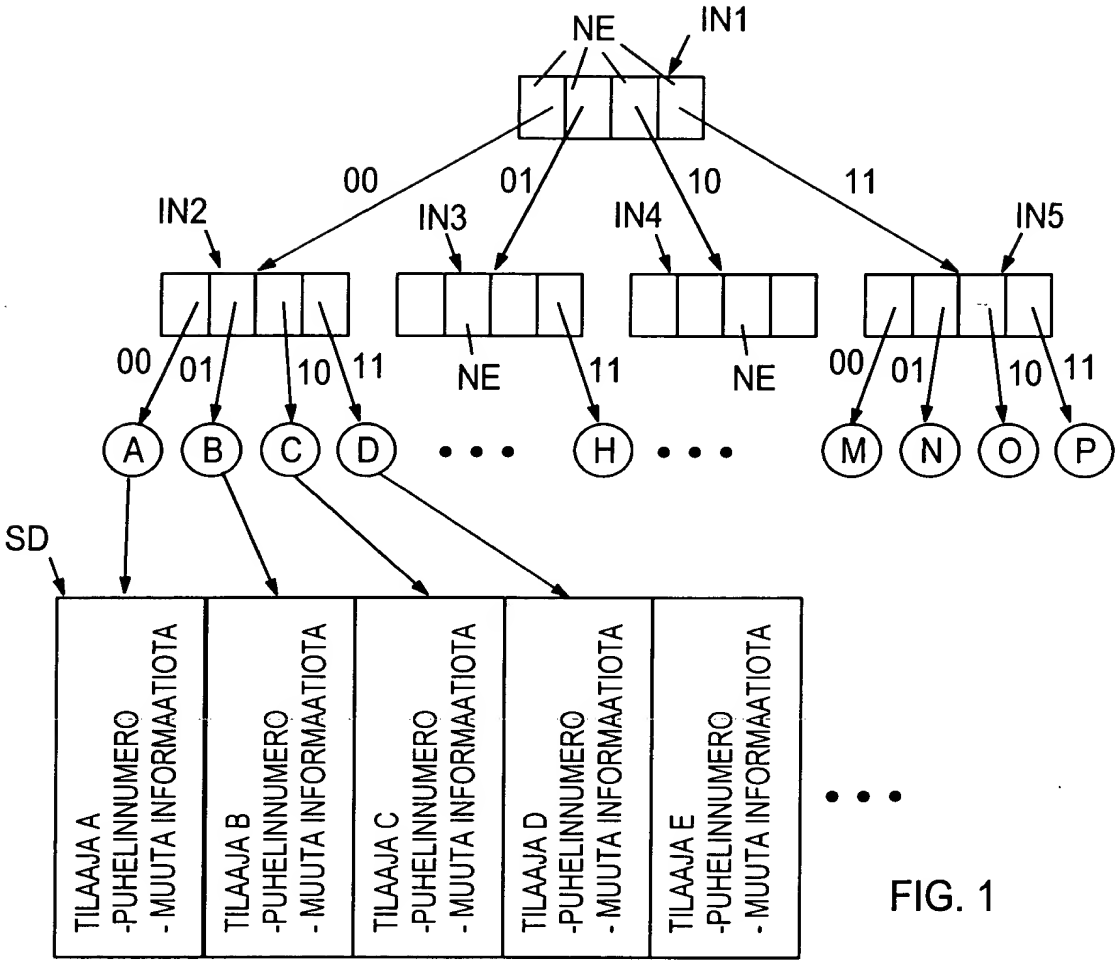


FIG. 1

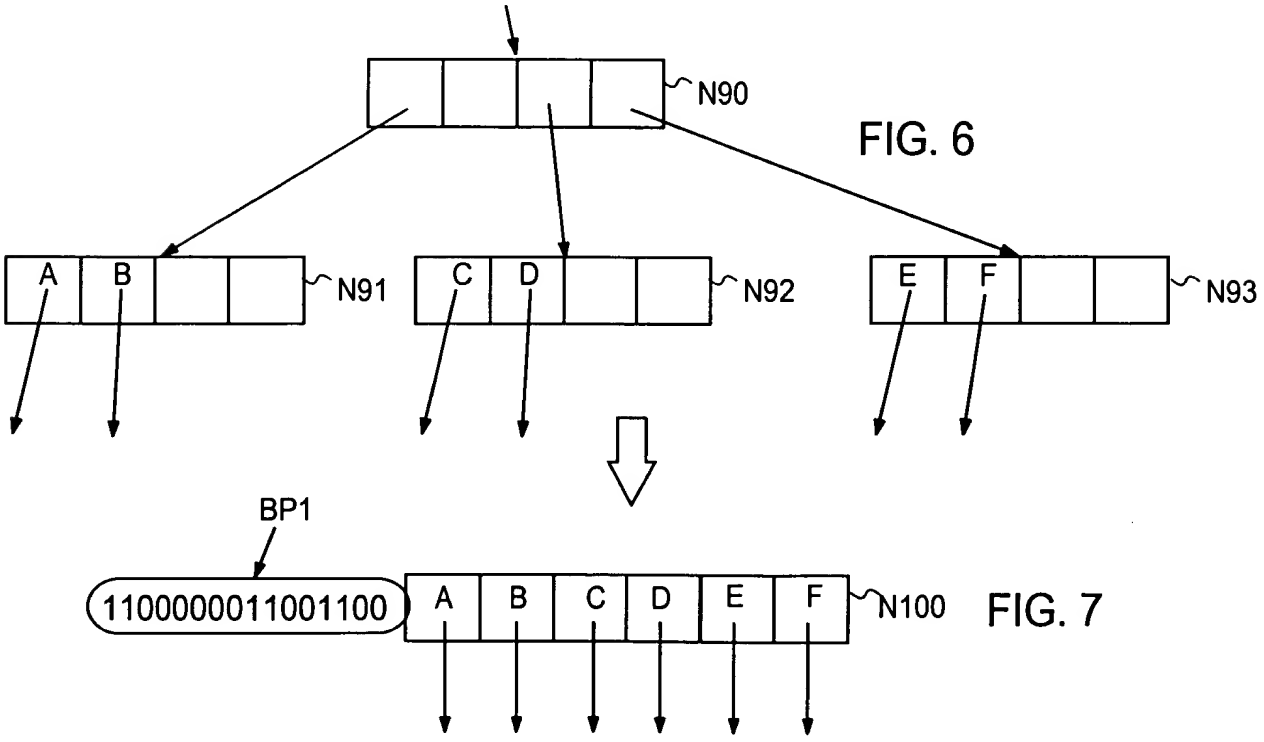


FIG. 6

FIG. 7

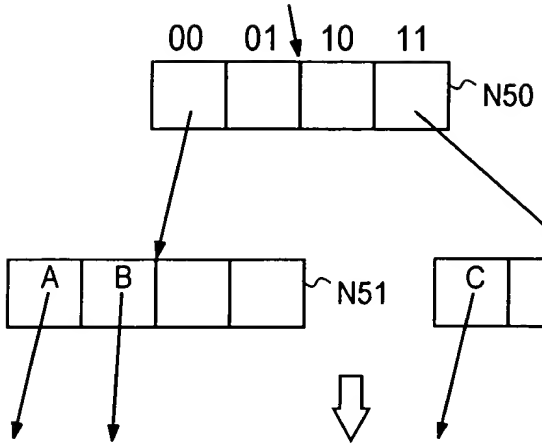


FIG. 2

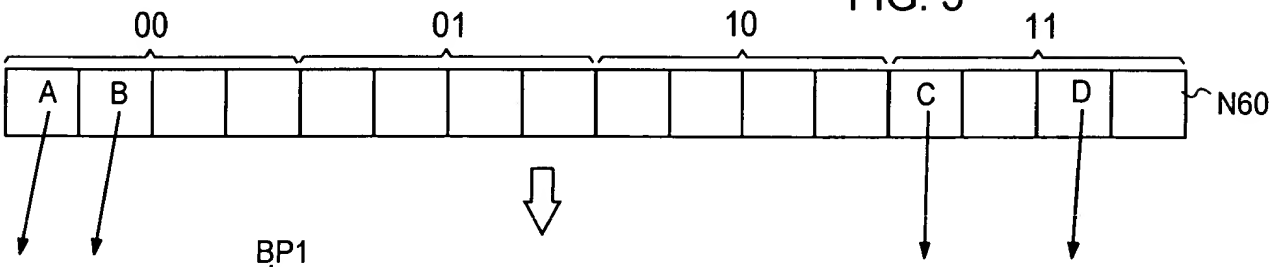


FIG. 3

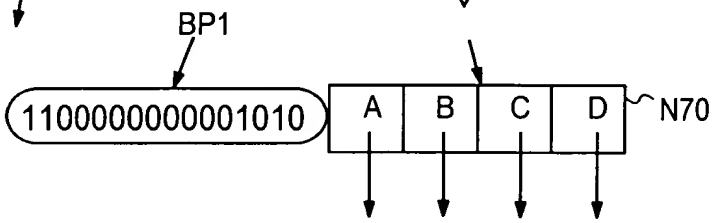


FIG. 4

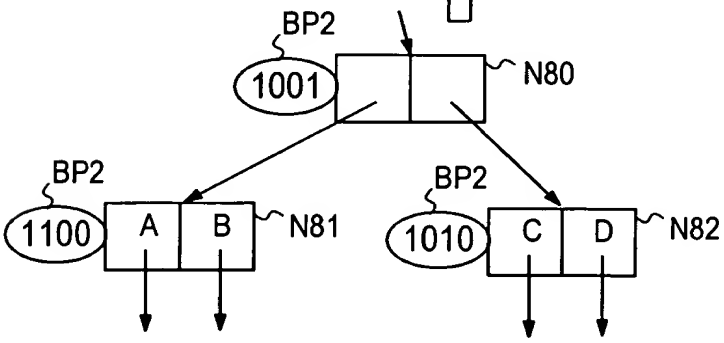


FIG. 5

FIG. 8

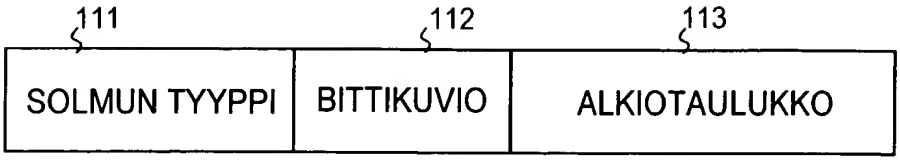
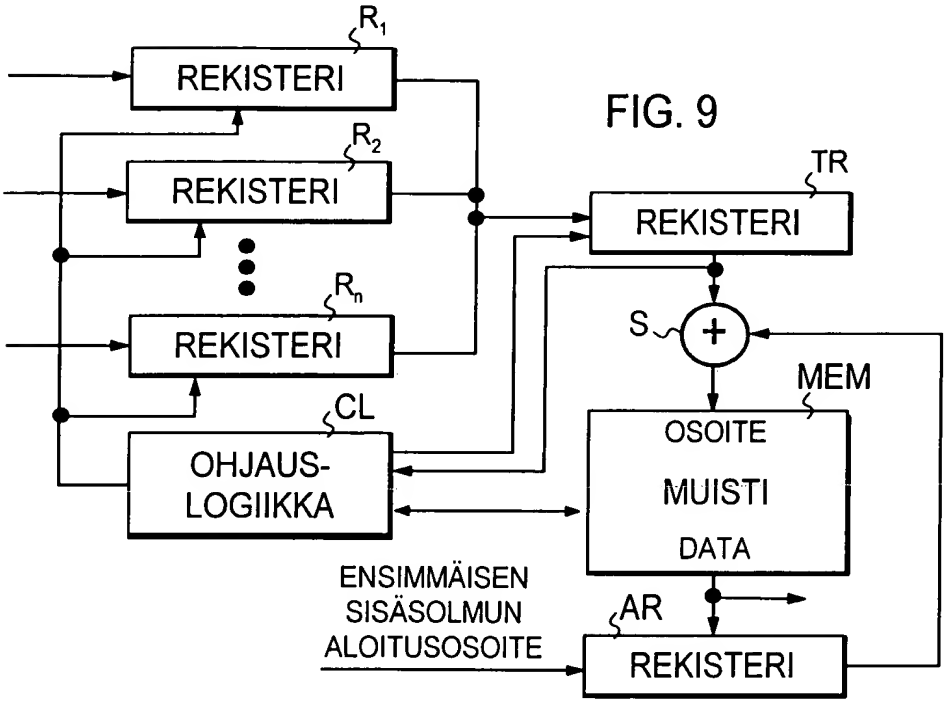

111
112
113
SOLMUN TYYPPI
BITTIKUVIO
ALKIOTAULUKKO

FIG. 9


R1
REKISTERI
R2
REKISTERI
Rn
REKISTERI
CL
OHJAUS-
LOGIIKKA
TR
REKISTERI
S
+
MEM
OSOITE
MUISTI
DATA
AR
REKISTERI
ENSIMMÄISEN
SISÄSOLMUN
ALOITUSOSOITE